

Generic Strategies for Chemical Space Exploration*

Jakob L. Andersen^{1,2}, Christoph Flamm²,
Daniel Merkle¹, and Peter F. Stadler²⁻⁷

¹ Department of Mathematics and Computer Science
University of Southern Denmark, Denmark

² Institute for Theoretical Chemistry, University of Vienna, Austria.

³ Bioinformatics Group, Department of Computer Science, and
Interdisciplinary Center for Bioinformatics, University of Leipzig, Germany.

⁴ Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany.

⁵ Fraunhofer Institute for Cell Therapy and Immunology, Leipzig, Germany.

⁶ Center for non-coding RNA in Technology and Health
University of Copenhagen, Denmark.

⁷ Santa Fe Institute, 1399 Hyde Park Rd, Santa Fe, NM 87501, USA

February 19, 2013

Abstract

Undirected labeled graphs and graph rewriting are natural models of chemical compounds and chemical reactions. This provides a basis for exploring spaces of molecules and computing reaction networks implicitly defined by graph grammars. Molecule graphs are connected, meaning that rewriting steps in general are many-to-many graph transformations. Chemical grammars are typically subject to combinatorial explosion, however, making it often infeasible to compute the underlying network by direct breadth-first expansion.

To alleviate this problem, we introduce here partial applications of rules as a basis for the efficient implementation of strategies that are not only well suited for exploration of chemistries defined by graph grammars, but that are also applicable in a general graph rewriting context as well. As showcases, we explore a complex chemistry based on the Diels-Alder reaction to explore specific subspaces of the molecular space. As a non-chemical application we use the framework of exploration strategies to model an abstract graph rewriting problem to construct high-level transformations that cannot be directly represented the Double-Pushout formalism starting from simple DPO transformation rules.

1 Introduction

The *structural formulae* of chemical compounds are graphs that represent the connectivity and mutual arrangements of the atoms. Atom types are given as vertex labels, while edges represent bond types. At this level of modelling, chemical reactions are naturally represented as graph transformations. Chemical reactions are explained and categorized in terms of *reaction mechanisms* that encapsulate the local changes of chemical bonds. In the formal framework of graph grammars, reaction mechanisms correspond to the productions (rules). Because of this conceptual alignment between chemistry and graph grammars, a variety of artificial chemistry models of different degree of chemical realism have been devised on this basis [6]. Of course, these purely combinatorial models of chemistry have their limitation. Deliberately disregarding the spatial embedding of molecules they cannot capture many aspects of stereochemistry and they are restricted to (over)simplified models of reactions energies and reaction kinetics. Graph grammar models are nevertheless of practical interest when the task is to explore large areas of chemical spaces and they provide a means of analyzing regularities in very large reaction networks.

*This work was supported in part by the Volkswagen Stiftung proj. no. I/82719, the COST-Action CM0703 "Systems Chemistry", and the Danish Council for Independent Research, Natural Sciences.

Of course there exist several graph rewriting tools, see for example [9] for an overview. The areas of application include model checking and verification, proof representation, and modeling control flow of programs among many others. A strategy language to control the application of graph rewriting rules has been presented in [8] for PORGY [2, 17]. In a chemical setting however, a strategy framework for exploring chemical spaces has very different needs. An application of a graph grammar rule might merge and split graphs, i.e., a strategy framework needs a chemically motivated component handling. Decisions on how to expand the space are usually heavily influenced by chemical properties or additional data sources. Furthermore, the goal of an analysis might also be motivated by a chemical question like the detection of chemical subspaces or finding specific chemical transformation patterns. The big need for a much more systematic exploration of chemical spaces has been identified also in chemistry and is discussed for example in [7].

The outline of the paper is as follows. The formal framework including the Double Pushout Approach is introduced in Section 2. In our chemical setting partial rule applications will be used which are described in Section 3. In Section 4 general strategies will be introduced. In Section 5 we will briefly comment on the implementation of the strategy framework. We will apply it to a complex chemical setting, namely the Diels-Alder reaction, and furthermore show results for an abstract graph rewriting problem in order to illustrate the concept of weakly connected subspaces. Results for these two settings are given in Section 6, and we conclude with Section 7.

2 Formal Framework

2.1 Chemical Graph Rewriting with the Double Pushout Approach

Molecules are always represented by connected graphs. Chemical reactions, however, more often than not, involve two or more interacting molecules as their “input” (educts) and there is no guarantee that the “output” (products) is connected. Thus we have to consider graph transformations that operate on not necessarily connected graphs. More precisely, we regard a graph G here as a multiset $\{g_1, g_2, \dots, g_{\#G}\}$ of its $\#G$ connected components. All graphs are simple. Double and triple bonds are viewed as edge labels rather than multiple edges.

Several abstract formalisms for graph transformation have been explored in the literature, see e.g., [18] for a detailed introduction. We found that the so-called Double Pushout (DPO) approach provides the most intuitive direct encoding of chemical reactions and the closest connection to the language of chemistry. A DPO transformation rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ consists of three graphs L , R and K known as the left, right and context graph, respectively, and two graph morphisms l and r that determine how the context is embedded in the left and the right graph. The rule p can be applied to a graph G if the left graph L can be found in G and some additional consistency conditions are satisfied. This is modeled by the requirement that there is a *matching morphism* $m : L \rightarrow G$ that describe how L is contained in G . Intuitively, the copy of L is replaced within G by R in such a way that the context K is left intact, resulting in the transformed graph H . This operation, the derivation $G \xrightarrow{p,m} H$, is described in the framework of category theory by the requirement that the following commutative diagram exists:

$$\begin{array}{ccccc} L & \xleftarrow{l} & K & \xrightarrow{r} & R \\ m \downarrow & & d \downarrow & & n \downarrow \\ G & \xleftarrow{\quad} & D & \xrightarrow{\quad} & H \end{array} \quad (1)$$

The derivation $G \xrightarrow{p,m} H$ implicitly define the intermediary graph D and the result graph H as well as morphisms $d : K \rightarrow D$ and $n : R \rightarrow H$ that fix how the context and the right graph of the rule are embedded in the intermediary and the result graph, respectively. In terms of molecules (connected components) we can write $\{g_1, g_2, \dots, g_{\#G}\} \Rightarrow \{h_1, h_2, \dots, h_{\#H}\}$.

In applications to modeling chemistry, several additional requirements must be satisfied. Conservation of mass and atom types dictates that the restrictions of r and l to the vertex sets (atoms) are bijective. Furthermore, m (and by extension d and n) are subgraph isomorphisms and hence injective. We note in passing that this guarantees the existence of a bijection $a : V(G) \rightarrow V(H)$ known as the *atom mapping*. In the DPO formalism, furthermore, the existence of an inverse production $p^{-1} = (L \xleftarrow{l} K \xrightarrow{r} R)$, corresponding to the reverse chemical reaction, is guaranteed. Some more basic properties of chemical graph grammars can be found in [1]. Fig. 1 shows an example of a chemical derivation.

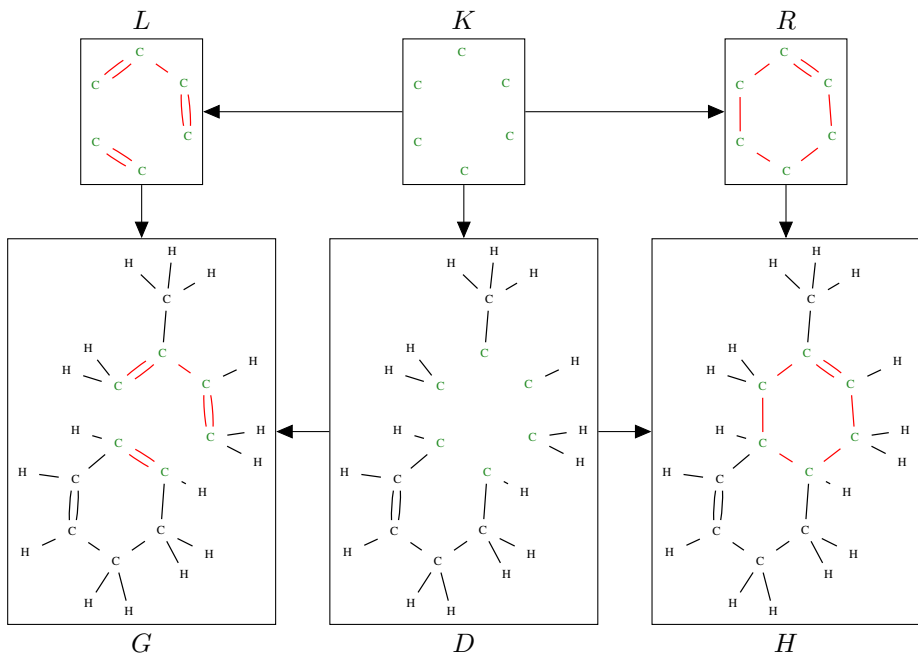


Figure 1: Example of a chemical derivation from Cyclohexadiene and Isoprene using a Diels-Alder transformation. The edges changed by the transformation is shown in red and the vertices from K are shown in green. Note that edges shown in parallel are in the underlying graphs a single edge with a special label to encode a specific chemical bond.

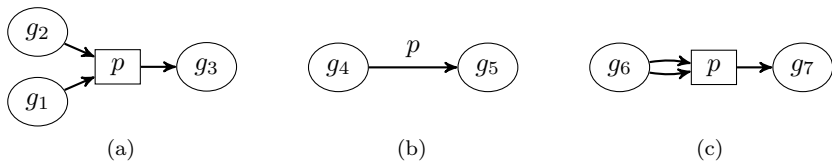


Figure 2: A bipartite graph notation, in which the production itself is drawn as a special type of intermediate vertex, is used in most cases; (a), $\{g_1, g_2\} \xrightarrow{p} g_3$. We only make an exception for 1-to-1 transformations (isomerization reactions); (b), $g_4 \xrightarrow{p} g_5$. Multiplicities are indicated by multiple arcs; (c), $\{g_6, g_6\} \xrightarrow{p} g_7$.

2.2 Proper Derivations

Consider a valid derivation $\{g_1, g_2\} \xrightarrow{p, m} \{h_1, h_2\}$ and an arbitrary graph g' . Clearly, the derivation $\{g_1, g_2, g'\} \xrightarrow{p, m} \{h_1, h_2, g'\}$ is also valid because the images of m and n are contained in $\{g_1, g_2\}$ and $\{h_1, h_2\}$, respectively. The graph g' is irrelevant for the transformation. We call a derivation $G \xrightarrow{p, m} H$ *proper* if $\text{img } m \cap g_i \neq \emptyset$ for all $g_i \in G$. It is not hard to see that the inverse of a proper derivation is again proper.

Throughout the following sections we will assume every derivation to be proper, unless otherwise stated.

2.3 Derivation Graphs

Chemical reaction networks can be represented as directed (multi)hypergraphs whose vertices are the molecules of the “chemical universe” under consideration and whose hyperedges represent chemical reactions [21]. Here, it is important to consider hyperedges as multisets to accommodate the stoichiometric coefficients, i.e., the multiplicities in which molecules enter a chemical reaction such as $2H_2 + O_2 \rightarrow 2H_2O$. Such networks can be constructed from experimentally observed data. An example is the Network of Organic Chemistry (NOC) [4, 10, 11], which shows a non-trivial organization concentrated around a core region of about 300 synthetically important building blocks and industrial compounds. Metabolic networks consist of the enzymatically catalyzed reactions constituting the chemical basis of modern life forms. They are available from dedicated databases, see e.g., [13].

In the framework of graph grammar models, an analogous *derivation graph* can be defined. Its vertex set consists of the connected labeled graphs \mathfrak{G} that represent the molecules. Directed hyperedges connect the multisets $G \subseteq \mathfrak{G}$ and $H \subseteq \mathfrak{G}$ only if there is a proper derivation $G \xrightarrow{p, m} H$. The conventions for visualizing

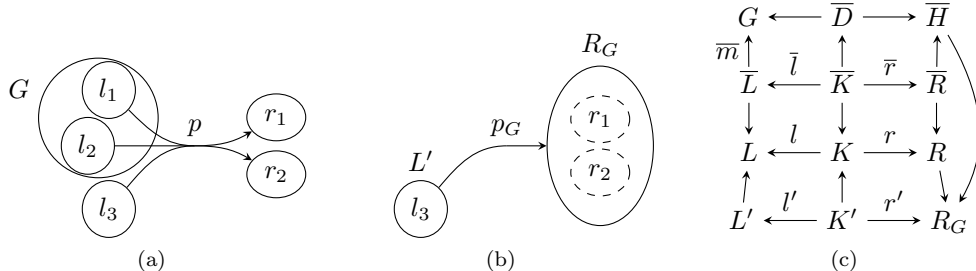


Figure 3: Partial application of some rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ to a graph G , with $L = \{l_1, l_2, l_3\}$ and $R = \{r_1, r_2\}$. The partial application is done through a partial matching morphism $\bar{m} : \bar{L} \rightarrow G$ with $\bar{L} = \{l_1, l_2\}$. The application results in a new rule, $p_G = (L' \xleftarrow{l'} K' \xrightarrow{r'} R_G)$ with $L' = \{l_3\}$, for which R is a subgraph of R_G . The transformed graph of G , called \bar{H} , is also a subgraph of R_G . Fig. (c) is the diagram of subgraph relations for a general partial rule application.

hyperedges adhere to the three examples in Fig. 2.

3 Transformation by Partial Rule Application

The core strategy to expand the underlying derivation graph is the discovery of new graphs by means of proper derivations implied by the direct application of rules. Given a rule $p = (L \leftarrow K \rightarrow R)$ and a set of graphs \mathfrak{U} , the task is to find all proper derivations $G \xRightarrow{p} H, G \subseteq \mathfrak{U}$ where G and H are multisets of graphs. This can be done by a testing of all k -multisubsets of \mathfrak{U} for all $1 \leq k \leq \#L$. Since nearly all chemical reactions are mono-molecular or bi-molecular, we can restrict ourselves to $\#L \leq 2$, at least when elementary reactions are of primary interest. Still, the number of multisets is $O(|\mathfrak{U}|^2)$. In the worst case, all unique multisets may give successful transformations, often leading to a combinatorial explosion that quickly becomes unmanageable. In the following section we show that a more detailed control of the multisets that are considered for transformation is desirable.

The key concept is partial rule composition [1], i.e., the binding of graphs to rules, resulting in partial rules that can be applied more efficiently in an exploration strategy. The idea is analogous to partial evaluation of functions by binding some of the variables. Full graph transformations are computed as repeated partial rule application in this framework. For the sake of brevity, we only sketch the idea here and omit a complete formal definition of partial rules.

A partial rule application of a rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ with $L = \{l_1, l_2, \dots, l_{\#L}\}$ to a graph G , is a generalization of a full transformation of G in which only some but not all components of L do not match G . Thus L is partitioned into the matching part $\bar{L} \neq \emptyset$ and the non-matching remainder L' . The restriction \bar{l} of $l : K \rightarrow L$ to the pre-image \bar{K} of \bar{L} defines the partial transformation rule $\bar{p} = (\bar{L} \xleftarrow{\bar{l}} \bar{K} \xrightarrow{\bar{r}} \bar{R})$. Using the restricted matching morphism $\bar{m} : \bar{L} \rightarrow G$ it can be applied to G resulting in graph \bar{H} . The remainder L' of L gives rise to a new rule $p_G = (L' \xleftarrow{l'} K' \xrightarrow{r'} R_G)$ whose right graph consists of the transformed version of G as well the original right graph R , i.e., it contains both \bar{H} and R as subgraphs. A formal, diagrammatic representation is given in Fig. 3c. An abstract partial application is shown in Fig. 3a and 3b.

Given a not necessarily connected graph G and DPO transformation rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$, our task is to construct all partial rules obtainable by binding G to p . These partial rules can then be applied to further graphs, allowing for more efficient exploration strategies. The following algorithm enumerates these partial rules:

1. For all $l_i \in L$ find the set of all subgraph isomorphisms of l_i to G . That is, find $M_i = \{m \mid m : l_i \rightarrow G \text{ is a subgraph isomorphism}\}$ for $1 \leq i \leq \#L$.
2. For all nonempty subsets \bar{L} of L , construct all partial matching morphisms, \bar{m} , by merging morphisms from each $M_j, l_j \in \bar{L}$. Note, that each \bar{m} must be injective.
3. For each partial matching morphism, \bar{m} , apply p to G with \bar{m} to obtain a new rule $p_G = (L' \xleftarrow{l'} K' \xrightarrow{r'} R_G)$.

The partial matching morphisms constructed from considering $\bar{L} = L$ are actually full matching morphisms, and so the resulting rule has $L' = K' = \emptyset$. In this case p_G represents the creation of R_G from an empty

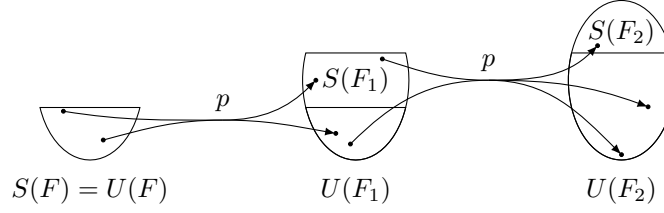


Figure 4: Illustration of the evaluation of $F_2 = p(F_1)$ for $F_1 = p(F)$ and some set of graphs, $S(F) = U(F) = \mathfrak{U}$. Each derivation must use at least one graph from the input subset. Two abstract derivations are shown with the endpoints indicating in which sets the graphs are.

graph, and $G \xrightarrow{p, \overline{m}} R_G$ is a valid derivation. If G is connected, the derivation will additionally be proper.

In the following section we will regard a rule p as a function on sets of graphs, defined provisionally as:

$$p(\mathfrak{U}) = \mathfrak{U} \cup \bigcup_{\substack{G \xrightarrow{p} H \\ G \subseteq \mathfrak{U}}} H$$

That is, the result of applying p to a set of graphs, \mathfrak{U} , is \mathfrak{U} itself along with all graphs derivable from \mathfrak{U} using p .

3.1 Complex Graph States

Consider the problem of applying a rule p twice to a set of graphs \mathfrak{U} . That is, finding $\mathfrak{U}_2 = p(\mathfrak{U}_1)$ for $\mathfrak{U}_1 = p(\mathfrak{U})$. By our definition of rule application we have $\mathfrak{U} \subseteq \mathfrak{U}_1$, so when the algorithm described above is used for evaluating $p(\mathfrak{U}_1)$ it will find not only new derivations but also all derivations found when evaluating $p(\mathfrak{U})$. We therefore use a more complex state than simply sets of graphs. A *graph state* F is defined as a pair of ordered sets of graphs $(\mathcal{U}, \mathcal{S})$ with $\mathcal{S} \subseteq \mathcal{U}$. The elements, \mathcal{U} and \mathcal{S} , will be referred to also as $U(F)$ and $S(F)$ respectively, where U and S are functions on the graph state. In the following we will denote $U(F)$ as the *universe* of the graph state F and $S(F)$ as the *subset* of the state. The order of graphs in the subset and in the universe is independent and is arbitrary unless otherwise stated.

We define the application of a rule p to a graph state F in the following manner. Let H' be all connected graphs derivable from $U(F)$ with p such that at least one graph from $S(U)$ is being transformed in each derivation:

$$H' = \{h \in H \mid G \xrightarrow{p} H : G \subseteq U(F) \wedge G \cap S(F) \neq \emptyset\} \quad (2)$$

The result $F' = p(F)$ is such that

$$\begin{aligned} U(F') &= U(F) \cup H' \\ S(F') &= H' \setminus U(F) \end{aligned} \quad (3)$$

That is, the resulting universe contains the input universe and all derived graphs, and the resulting subset contains all new graphs which was not known before. The removal of known graphs from the output subset is motivated by the goal of exploring the underlying network of derivations. However, this specific behaviour is not always desired so an alternate definition can be used (see Section 4.7).

With the definition above we rewrite our initial example as; find $F_2 = p(F_1)$ for $F_1 = p(F)$ and $S(F) = U(F) = \mathfrak{U}$. The application $p(F_1)$ can now only discover derivations with at least one graph from $S(F_1)$, which by definition contains only new graphs. Therefore, only new derivations are found. Fig. 4 contains a visualization of the example.

The implementation utilizes the algorithm for transformation by first partially applying the rule to the subset of the input state, and then afterwards the full universe.

4 Strategies

The previous section described how a rule p is applied to a state F to calculate a new state F' , and motivated this by the example of composition of rule application, $F' = p(p(F))$. Using the definition of a graph state, we generalize the interface for rule application into general strategies. A strategy is simply any function Q from and to the set of graph states.

In the following we introduce core strategies defined in the framework. Most of the strategies are parameterized, which we will note with brackets around these parameters. The application of a strategy Q with some fixed parameter, n , to a graph state F is thus denoted as $Q[n](F)$.

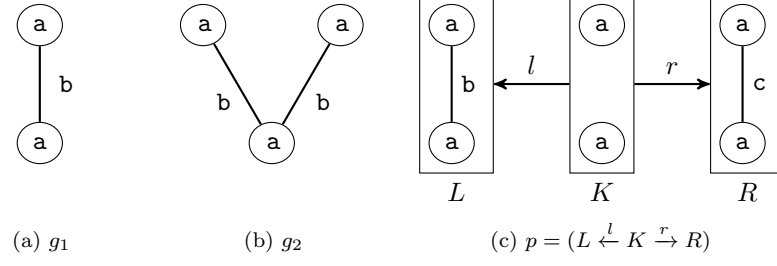


Figure 5: Graphs and transformation rule for the example of the semantics of revive strategies.

4.1 Parallel

A parallel strategy is defined in terms of a set of substrategies, $\{Q_1, Q_2, \dots, Q_n\}$. The result of applying a parallel strategy is the union of the results from applying the individual substrategies:

$$\begin{aligned}
 F' &= \text{parallel}[\{Q_1, Q_2, \dots, Q_n\}](F) \\
 U(F') &= \bigcup_{1 \leq i \leq n} U(Q_i(F)) \\
 S(F') &= \bigcup_{1 \leq i \leq n} S(Q_i(F))
 \end{aligned}$$

The order of the resulting universe and subset is arbitrary.

A simple example of the use of a parallel strategy is the application of multiple rules to the same set of graphs.

4.2 Sequence

A sequence strategy, Q , is a composition of a list of substrategies, Q_1, Q_2, \dots, Q_n :

$$Q(F) = Q_n(\dots(Q_2(Q_1(F))))$$

To increase left-to-right readability of sequence strategies, we will use the notation $Q = Q_1 \rightarrow Q_2 \rightarrow \dots \rightarrow Q_n$. Additionally, if $Q_1 = Q_2 = \dots = Q_n = Q'$, we may use the normal notation for powers of functions, $Q = Q'^n$, for the sequence.

4.3 Repetition

The sequencing strategy only allows composition of a fixed number of strategies, whereas the repetition strategy is used to compose a single strategy with itself many times.

A repetition strategy, Q , is parameterized by a non-negative integer, n , and an inner strategy Q' . The inner strategy is composed with itself until the graph state reaches a fixed point or its subset is empty, however at most n times:

$$\begin{aligned}
 Q &= \text{repeat}[Q', n] = Q'^k \\
 k &= \min i \in \{0, 1, \dots, n\}, \text{ such that } Q'^i(F) = Q'^{i+1}(F) \vee S(Q'^{i+1}(F)) = \emptyset \vee i = n
 \end{aligned}$$

This means that if the graph state reaches a fixed point then that graph state is returned, and if the subset of the state becomes empty then the *previous* state is returned. We motivate this condition of a non-empty subset of a produced graph state by our definition of rule application, which requires at least one graph from the subset. By returning the last graph state with non-empty subset the repetition strategy can be used as a precomputation in a sequence to find a kind of closure under some inner strategy.

Note that for $k = 0$ the strategy becomes the identity strategy. If k is set large enough to not limit the repetition, we call it unbounded repetition, and write it as $Q = \text{repeat}[Q']$.

4.4 Revive

The strategy framework is primarily aimed at generating a network representing derivations. However, it may be useful to use the strategies directly as functions on graphs which means that repetition strategies may not work as expected.

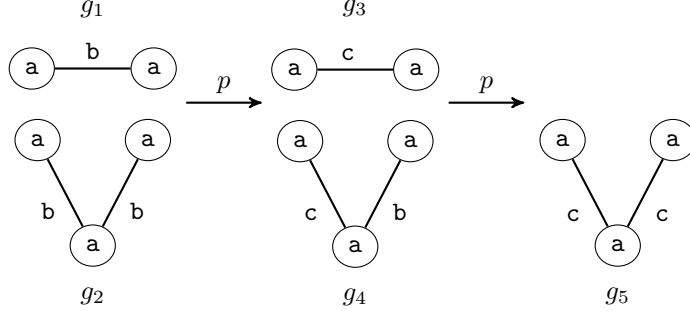


Figure 6: Illustration of the application of **repeat**[p] to F with $S(F) = U(F) = \{g_1, g_2\}$. Only the subset of the graph states are shown. The first application of p results in two new graphs, g_3 and g_4 , but as p can only be applied to g_4 the final subset is only a single graph, g_5 , instead of both g_3 and g_5 .

Consider the following problem. Two graphs, g_1 and g_2 and the transformation rule p , as illustrated in Fig. 5 are given. We wish to develop a strategy to transform all edge labels using rule p , with the intend to use this strategy as a precomputation for a subsequent strategy. That is, the result of the strategy must contain the completely transformed graphs in the subset. From this specification we first try the strategy $Q = \mathbf{repeat}[p]$ applied to the graph state F with $S(F) = U(F) = \{g_1, g_2\}$, which may be the most intuitive approach. However, it does not give the intended result. This is illustrated in Fig. 6.

The intention of the revive strategy is to provide a mechanism to solve this problem. Such a strategy, Q , is defined in terms of an inner strategy, Q' , and is written as $Q = \mathbf{revive}[Q']$. A rule strategy creates a (possibly empty) set of derivations, and by extension a strategy creates derivations. We say that a graph g is consumed in strategy Q if Q creates a derivation $G \Rightarrow H$ with $g \in G$. With this we define the revive strategy as:

$$\begin{aligned} F' &= \mathbf{revive}[Q'](F) \\ U(F') &= U(Q'(F)) \\ S(F') &= S(Q'(F)) \cup \{g \in S(F) \mid g \in U(F') \wedge g \text{ is not consumed in } Q'\} \end{aligned}$$

That is, any graph from the input subset which is still in the output universe and was not consumed, will be added to the output subset.

The example problem can with the revive strategy be solved with $Q = \mathbf{repeat}[\mathbf{revive}[Q']]$.

4.5 Derivation Predicates

For the purpose of precise modeling and the problems with combinatorial explosion it is convenient to limit the possibilities of expansion. We define two variations of the concept of derivation predicates, which both introduces extra constraints in Eq. (2) to prune unwanted derivations. The strategies, **leftPredicate**[P, Q'] and **rightPredicate**[P, Q'], are both defined in terms of a predicate, P , on a transformation rule and a multiset of graphs, and an inner strategy, Q' . For a left predicate with P and Q' , each derivation $G \xRightarrow{p} H$ found by Q' must satisfy $P(p, G)$. In a right predicate it must be $P(p, H)$ which is true.

As example, given a strategy Q' we wish to produce only graphs with at most 42 vertices (atoms, in a chemical context). This can be specified with the following strategy:

$$\begin{aligned} Q &= \mathbf{rightPredicate}[P, Q'] \\ P(p, \{g_1, g_2, \dots, g_k\}) &\equiv \forall 1 \leq i \leq k : |V(g_i)| \leq 42 \end{aligned}$$

4.6 Filter, Sort, Take and Add

To facilitate more elaborate use of strategies in a functional style we define several strategies which correspond to functions on lists in other languages. As a graph state is composed of both a universe and a subset, all of these strategies are defined in two variations.

A filter strategy is parameterized by a predicate on a graph and a graph state:

$$\left. \begin{aligned} F' &= \mathbf{filterSubset}[P](F) \\ U(F') &= U(F) \\ S(F') &= \{g \in S(F) \mid P(g, F)\} \end{aligned} \right| \begin{aligned} F' &= \mathbf{filterUniverse}[P](F) \\ U(F') &= \{g \in U(F) \mid P(g, F)\} \\ S(F') &= \{g \in S(F) \mid P(g, F)\} \end{aligned}$$

A sorting strategy is parameterized with a predicate on two graphs and a graph state, used as a less-than operator in a stable sort of a list of graphs:

$$\begin{array}{l|l} F' = \text{sortSubset}[P](F) & F' = \text{sortUniverse}[P](F) \\ U(F') = U(F) & U(F') = \text{stableSort}[P](U(F)) \\ S(F') = \text{stableSort}[P](S(F)) & S(F') = S(F) \end{array}$$

The choice that the sorting algorithm must be stable is motivated by the desire to allow lexicographical sorting by sequencing several sorting strategies.

A take strategy is parameterized with a natural number:

$$\begin{array}{l|l} F' = \text{takeSubset}[n](F) & F' = \text{takeUniverse}[n](F) \\ k = \min\{n, |S(F)|\} & k = \min\{n, |U(F)|\} \\ U(F') = U(F) & U(F') = \{U(F)_1, U(F)_2, \dots, U(F)_k\} \\ S(F') = \{S(F)_1, S(F)_2, \dots, S(F)_k\} & S(F') = S(F) \cap U(F') \end{array}$$

An addition strategy appends a given set of graphs to either the universe and optionally also to the subset:

$$\begin{array}{l|l} F' = \text{addSubset}[\{g_1, g_2, \dots, g_n\}](F) & F' = \text{addUniverse}[\{g_1, g_2, \dots, g_n\}](F) \\ U(F') = U(F) \cup \{g_1, g_2, \dots, g_n\} & U(F') = U(F) \cup \{g_1, g_2, \dots, g_n\} \\ S(F') = S(F) \cup \{g_1, g_2, \dots, g_n\} & S(F') = S(F) \end{array}$$

An example usage of these strategies is procedure of ranking graphs according to some property, take the best n graphs for subsequence expansion, i.e:

$$Q' = \text{sortSubset}[P] \rightarrow \text{takeSubset}[n]$$

The addition strategies can be used both for injecting new graphs in the middle of a strategy, but we also find them convenient simply for uniform left-to-right writing of a strategy application. E.g., given a (large) strategy Q we wish to apply to the graph state F , we can write:

$$F' := \text{addUniverse}[U(F)] \rightarrow \text{addSubset}[S(F)] \rightarrow Q$$

with the interpretation $F' = Q(F)$.

4.7 Alternate Rule Application

The definition of rule application described previously is aimed at exploration of the underlying space of graphs. In particular the definition of the output subset of rule application, Eq. (3), manipulates the graph state such that already discovered graphs can not initiate another derivation. To facilitate the use of the strategy framework for more direct functional computation it is desired to let the resulting subset of rule application be all derived graphs, i.e., change Eq. (3) to $S(F') = \mathcal{H}$. We therefore introduce a strategy to change this behaviour for a given inner strategy. That is, the strategy $Q = \text{altRuleApp}[Q']$ evaluates as Q' , but with all rule applications in Q' using the alternate subset definition.

5 Implementation Remarks

The strategies are implemented in C++ as part of a library, to allow easy extension at the user level. Extensions can vary from simple graph state manipulating strategies to complete replacement of the underlying transformation formalism. The library is aimed at chemical graph transformation, with special optimization for molecules (e.g., use of canonical SMILES strings for graph isomorphism [19, 20]), but is not restricted to the domain of chemistry. The current implementation uses VF2[5] to find subgraph isomorphisms, and as a fall-back algorithm for isomorphism check for general graphs. Furthermore, the library utilizes data structures and procedures for molecule handling from the Graph Grammar Library (GGL) [3].

An interpreter for a custom language with graphs, rules and strategies as data types, is also implemented to allow easy development of expansion strategies.

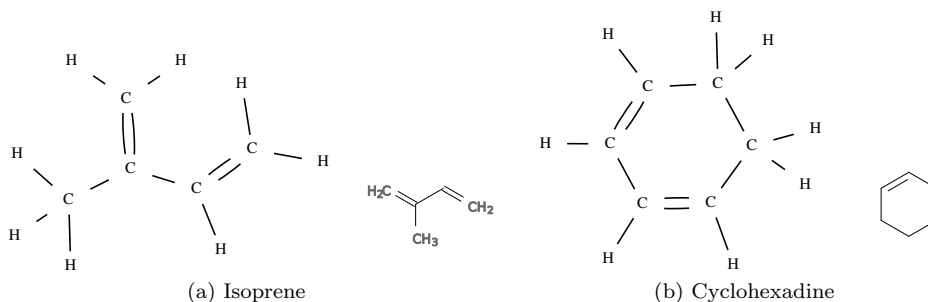


Figure 7: The starting molecules, (a) isoprene and (b) cyclohexadiene, for application of the Diels-Alder reaction. The molecules are shown in two versions; one with all vertices explicit and chemical interpretation of edge labels (left), and one version in standard chemical visualization.

6 Results

The Diels-Alder reaction is one of the most useful reaction in organic chemistry and has heavily influenced total synthesis in the last decades [16]. The explosion of the chemical space by applying this reaction several times will be biased by the strategy framework. In order to easily illustrate subspaces that are also expected to exist in a chemical setting, we will apply the strategy framework to a small puzzle game.

6.1 The Diels-Alder Reaction

As a small, but complex, chemistry we here use the Diels-Alder reaction with two starting molecules. The reaction is shown in an example derivation in Fig. 1, while the starting molecules, isoprene and cyclohexadiene, are shown in Fig. 7. Let $p = (L \leftarrow K \rightarrow R)$ be the transformation rule modeling the Diels-Alder reaction. The intention of the rule is that it is applied to two molecules, but this constraint is not encoded in the rule. We therefore first wrap p with a derivation predicate:

$$Q_p = \text{leftPredicate}[P, p] \quad P(p', G) \equiv \#G = 2$$

This means that all derivations $G \xrightarrow{p} H$ must have $|G| = 2$.

A generic breadth-first exploration of the chemical space can be done with the following strategy:

$$Q_{\text{BFS}} = \text{addSubset}[\{\text{isoprene}, \text{cyclohexadiene}\}] \rightarrow \text{repeat}[Q_p, n]$$

However, for $n = 4$ the strategy already discovers 825 new graphs through 1278 derivations.¹ The number of subgraph isomorphism queries throughout the evaluation is 74591. In Appendix A, Fig. 11 the resulting derivation graph for just $n = 2$ is shown.

We now decide to only look at the subspace of molecules which are derived by repeatedly merging molecules with isoprene, starting with cyclohexadiene. The following strategy implements this specification:

$$\begin{aligned} Q_{\text{subspace}} &= \text{addUniverse}[\{\text{isoprene}\}] \rightarrow \text{addSubset}[\{\text{cyclohexadiene}\}] \\ &\rightarrow \text{leftPredicate}[P_{\text{init}}, Q_p] \rightarrow \text{filterUniverse}[P_{\text{filter}}] \\ &\rightarrow \text{repeat}[Q_p, n] \end{aligned} \quad (4)$$

with

$$\begin{aligned} P_{\text{init}}(p', G) &\equiv G = \{\text{isoprene}, \text{cyclohexadiene}\} \\ P_{\text{filter}}(g, F) &\equiv g \neq \text{cyclohexadiene} \end{aligned}$$

This first computes all possible proper derivations $\{\text{isoprene}, \text{cyclohexadiene}\} \xrightarrow{p} H$, then removes cyclohexadiene from the graph state to prevent further derivations. In the end it uses breadth-first expansion for at most n steps. This strategy, with $n = 3$ (i.e., 4 expansion steps including the very specific first step) discovers only 165 new graphs through 236 derivations,² and uses 5524 subgraph isomorphism queries. The derivation graph with $n = 2$ is visualized in Fig. 8.

¹In this scenario we regard derivations which only differ in the matching morphism as duplicates. The evaluation of the strategy takes in the order of 10 seconds with a Intel® Core™ i5-2500K CPU (3.30GHz).

²— “ —, resp., 8 seconds

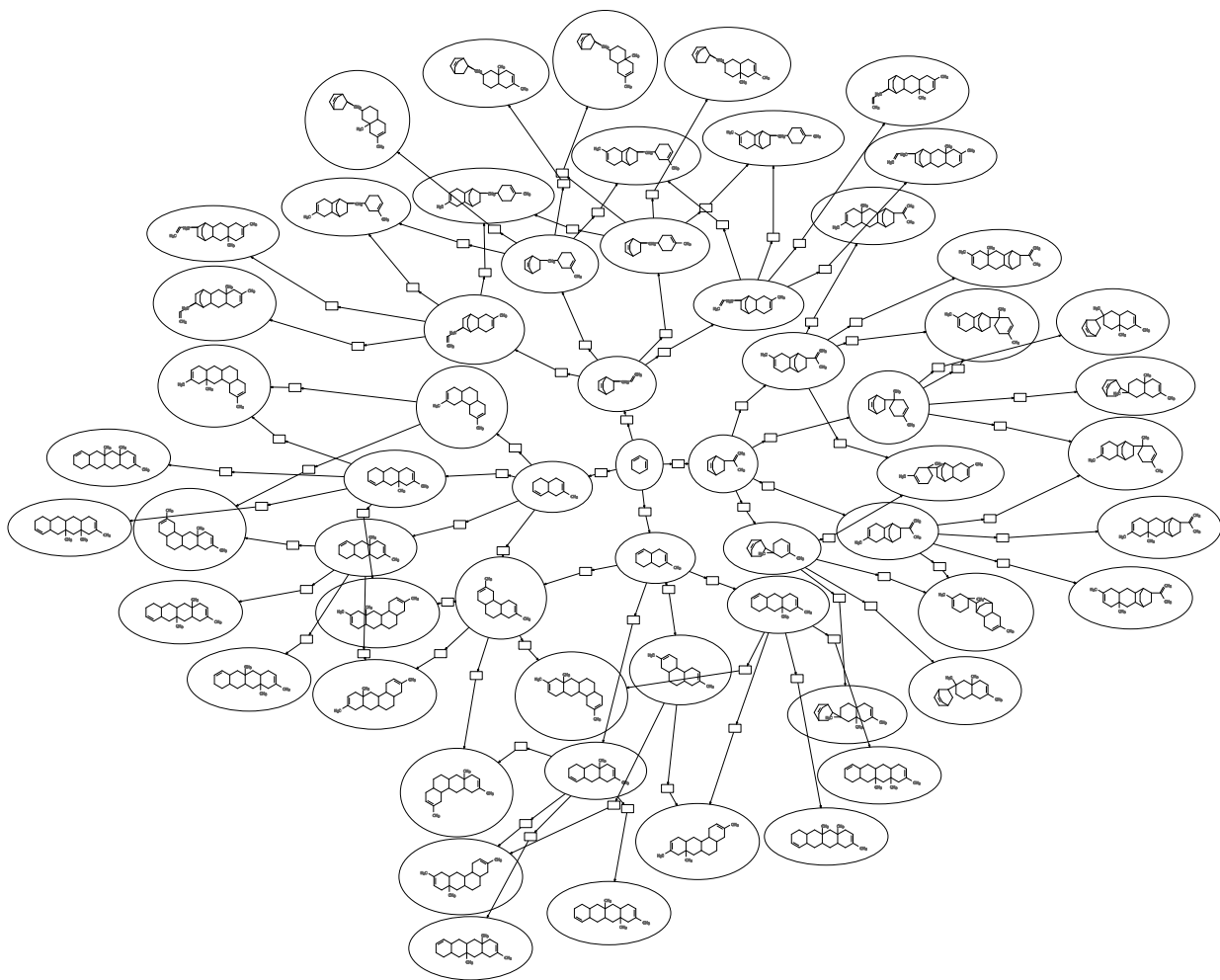


Figure 8: The derivation graph resulting from evaluating the expansion strategy Q_{subspace} , Eq. (4). To minimize clutter, the vertex with isoprene and the corresponding edges are not shown, although isoprene is involved in any reaction (the resulting chemical reaction network is a hypergraph).

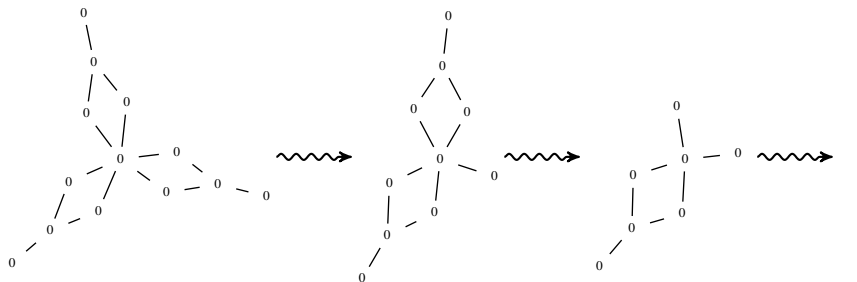


Figure 9: Level 1 of the Catalan game and the intermediary graphs during transformation to a graph with a single vertex.

6.2 Solving the Catalan Game

The Catalan game [12] is a puzzle game in which the player in each level is presented with a simple undirected graph without labels. The goal is to transform the graph into a single vertex using the following rewriting rule; given a vertex v with degree exactly 3, identify v with its neighbours and preserve simpleness of the graph by identifying parallel edges and deleting loops. Fig. 9 shows level 1 with the intermediary graphs towards the goal graph with a single vertex.

The transformation in the game can not be formulated as a single rule in the DPO formalism, because such rules must explicitly match the vertices and edges which are changed, while the Catalan transformation needs to change arbitrarily many edges. In the following we show how the strategies can be used to implement a move in the game, using only DPO rules.

Let g be the graph from some Catalan level, with all edge labels set to the empty string and all vertex labels set to the arbitrarily chosen label “0”. A high-level description of a move is:

1. Find a vertex v with at least 3 neighbours and mark it by changing the label to "A". Mark the 3 matched neighbours with the label "R".
2. If possible, find another fourth neighbour of v and mark v with "FAIL".
3. Discard all graphs with a vertex with the label "FAIL".
4. For all edges e with both end-vertices having label "R", remove e .
5. For all edges ur with u having label "0" and r having label "R", add uv if it does not exist already and then remove ur .
6. For all edges ur with u having label "0" and r having label "R", remove ur .
7. Remove all neighbours of v having label "R".
8. Unmark v by changing the label to "0".

Step 3 can be implemented with a filtering strategy while the other steps each require a transformation rule. The following strategy can be used to solve a level, in the sense that if a graph with a single vertex with label "0" is found, then a path to that graph is equivalent to a solution. The details of the transformation rules (mark, markForFail, removeInterR, reattachExternal, removeAttached, removeR and unmark) are shown in Appendix B.

$$Q_{\text{catalan}} = \text{addSubset}[\{g\}] \rightarrow \text{altRuleApp}[\text{repeat}[\text{mark} \rightarrow \text{revive}[\text{markForFail}] \rightarrow \text{filterUniverse}[P_{\text{fail}}] \rightarrow \text{repeat}[\text{revive}[\text{removeInterR}]] \rightarrow \text{repeat}[\text{revive}[\text{reattachExternal}]] \rightarrow \text{repeat}[\text{revive}[\text{removeAttached}]] \rightarrow \text{removeR} \rightarrow \text{unmark}]]]$$

$$P(g', F) \equiv \text{no vertex of } g' \text{ has the label "FAIL"}$$

With strategy Q_{catalan} all 56 levels of Catalan could be solved, all but one level took less than 10 minutes of computation time. Fig. 10b exemplarily shows the derivation graph created when executing the strategy with g encoding level 25 of the game, and Fig. 10a show the initial level graph. The resulting derivation graph is, in contrast to chemical reaction networks, not a hypergraph. However, the graph clearly illustrates subspaces that are connected via a small number of bridging edges. Such subspaces are also expected in chemical reaction networks.

7 Conclusions

We introduced here generic strategies for the systematic exploration of spaces of graphs. Our generative approaches use the Double Pushout formalism in order to derive new graphs. Since this task is of practical relevance in chemistry [7], we designed our framework and implementation with the aim of high efficiency in this particular domain of application. It is in no way restricted to this area, however. As an example we showed that our implementation of graph strategies can be used to treat high-level transformation rules on graphs that cannot be formulated as graph grammars with a finite number of rules since, e.g., the size of subgraph that is affected by the transformation is not bounded by the rule but only by the input graphs.

Performance was a particular focus of our work, although this has not been discussed in detail here. We use state-of-the-art subgraph isomorphism check methods and we heavily employ hashing techniques in order to check for graph isomorphisms; in order to infer proper derivations of new molecules with full or partial rule application we do *not* use a straightforward method to enumerate all possible left-hand-sides of derivations. Instead we employ partial rule applications, a method that shows theoretically as well as empirically a much better performance (a detailed discussion will be published elsewhere).

In order to analyze chemical reaction networks as created by our strategy framework, there exist several mathematical techniques that we plan to apply. Two of the most prominent ones are Flux Balance Analysis [14] and Elementary Mode Analysis [15]. Note, that these methods are usually not applied to dynamically created reaction networks as produced by our framework. We aim at detecting new well-defined chemical reaction pattern. Furthermore, we expect to identify highly connected subgraphs in chemical spaces, that are connected via a small number of bridging reaction, similar to our observation for the Catalan game.

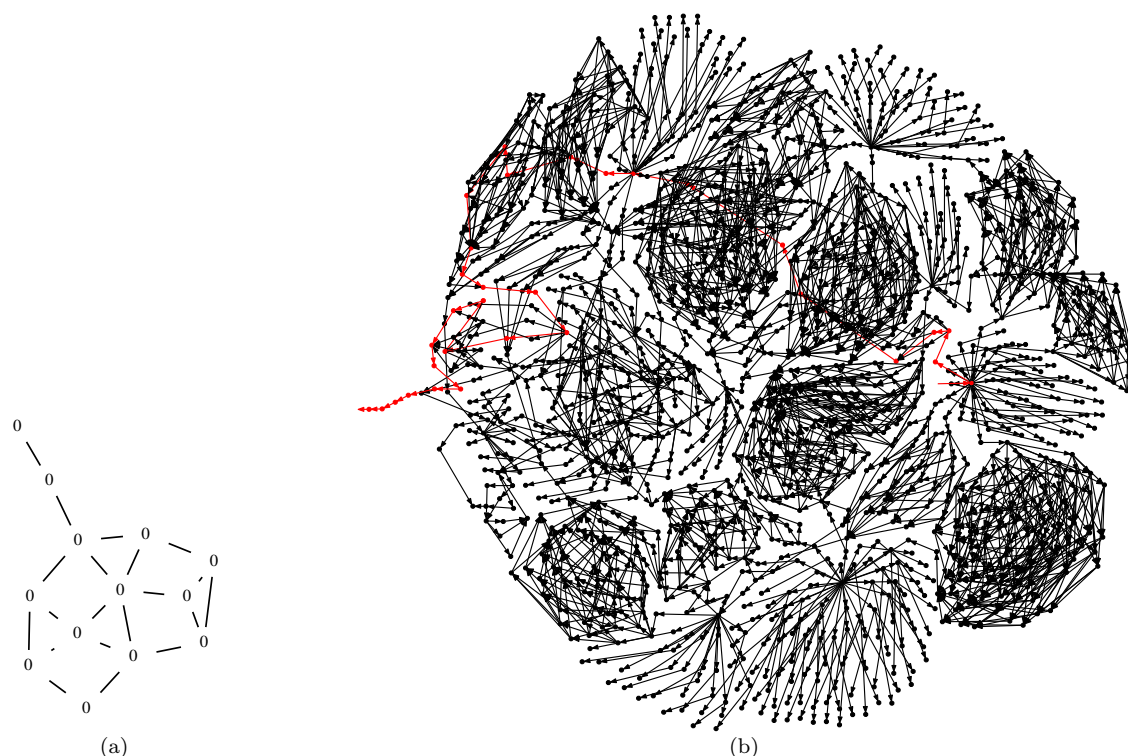


Figure 10: The derivation graph created during expansion of level 25 of the Catalan game. A path equivalent to a solution is highlighted.

References

- [1] J.L. Andersen, C. Flamm, D. Merkle, and P.F. Stadler. Inferring chemical reaction patterns using graph grammar rule composition. 2012. submitted.
- [2] O. Andrei, M. Fernández, H. Kirchner, G. Melançon, O. Namet, and B. Pinaud. PORGY: Strategy driven interactive transformation of graphs. In *In Proceedings of the 6th International Workshop on Computing with Terms and Graphs (TERMGRAPH 2011)*, volume 48 of *Electronic Proceedings in Theoretical Computer Science*, pages 54–68, 2011.
- [3] G. Benkő, C. Flamm, and P. F. Stadler. A graph-based toy model of chemistry. *J. Chem. Inf. Comput. Sci.*, 43(4):1085 – 1093, 2003.
- [4] K.J.M. Bishop, R. Klajn, and B.A. Grzybowski. The core and most useful molecules in organic chemistry. *Angew. Chem. Int. Ed.*, 45:5348–5354, 2006.
- [5] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367, 2004.
- [6] P. Dittrich, J. Ziegler, and W. Banzhaf. Artificial chemistries - a review. *Artificial life*, 7(3):225–275, 2001.
- [7] M. Dow, M. Fisher, T. James, F. Marchetti, and A. Nelson. Towards the systematic exploration of chemical space. *Org. Biomol. Chem.*, 10:17–28, 2012.
- [8] M. Fernández, H. Kirchner, and O. Namet. A strategy language for graph rewriting. In *Proceedings of the 21st International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2011)*, volume 7225 of *Lecture Notes in Computer Science*, pages 173–188, 2012.
- [9] M. Fernández and O. Namet. Strategic programming on graph rewriting systems. In *Proceedings of the 1st International Workshop on Strategies in Rewriting, Proving, and Programming (IWS 2010)*, volume 44 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–20, 2010.
- [10] M. Fialkowski, K.J.M. Bishop, V.A. Chubukov, C.J. Campbell, and B.A. Grzybowski. Architecture and evolution of organic chemistry. *Angew. Chem. Int. Ed.*, 44:7263–7269, 2005.

- [11] B.A. Grzybowski, K.J.M. Bishop, B. Kowalczyk, and C.E. Wilmer. The 'wired' universe of organic chemistry. *Nature Chemistry*, 1:31–36, 2009.
- [12] increpare games. Catalan, accessed 04. Feb. 2013. <http://www.increpare.com/2011/01/catalan/>.
- [13] P.D. Karp and R. Caspi. A survey of metabolic databases emphasizing the MetaCyc family. *Arch. Toxicol.*, 85:1015–1033, 2011.
- [14] K. J. Kauffman, P. Prakash, and J. S. Edwards. Advances in flux balance analysis. *Curr. Opin. Biotechnol.*, 14(5):491 – 496, 2003.
- [15] S. Klamt and J. Stelling. Two approaches for metabolic pathway analysis? *Trends Biotechnol.*, 21(2):64 – 69, 2003.
- [16] K.C. Nicolaou, S.A. Snyder, and G. Montagnon, T. amd Vassilikogiannakis. The Diels-Alder Reaction in total synthesis. *Angew. Chem. Int. Ed.*, 41:1668–1698, 2002.
- [17] Bruno Pinaud, Guy Melançon, and Jonathan Dubois. PORGY: A visual graph rewriting environment for complex systems. *Comput. Graph. Forum*, 31(3), 2012.
- [18] G. Rozenberg and H. Ehrig. *Handbook of graph grammars and computing by graph transformation*, volume 1. World Scientific, Singapore, 1997.
- [19] D. Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28(1):31 – 36, 1988.
- [20] D. Weininger, A. Weininger, and J. L. Weininger. SMILES 2. Algorithm for Generation of Unique SMILES Notation. *J. Chem. Inf. Comput. Sci.*, 29(2):97 – 101, 1989.
- [21] A. V. Zeigarnik. On hypercycles and hypercircuits in hypergraphs. In P. Hansen, P. W. Fowler, and M. Zheng, editors, *Discrete Mathematical Chemistry*, volume 51 of *DIMACS series in discrete mathematics and theoretical computer science*, pages 377–383. American Mathematical Society, Providence, RI, 2000.

A Additional Diels-Alder Chemistry Figure

Fig. 11 shows the derivation graph obtained from the breadth-first expansion of the Diels-Alder chemistry. The number of expansion steps is only 2.

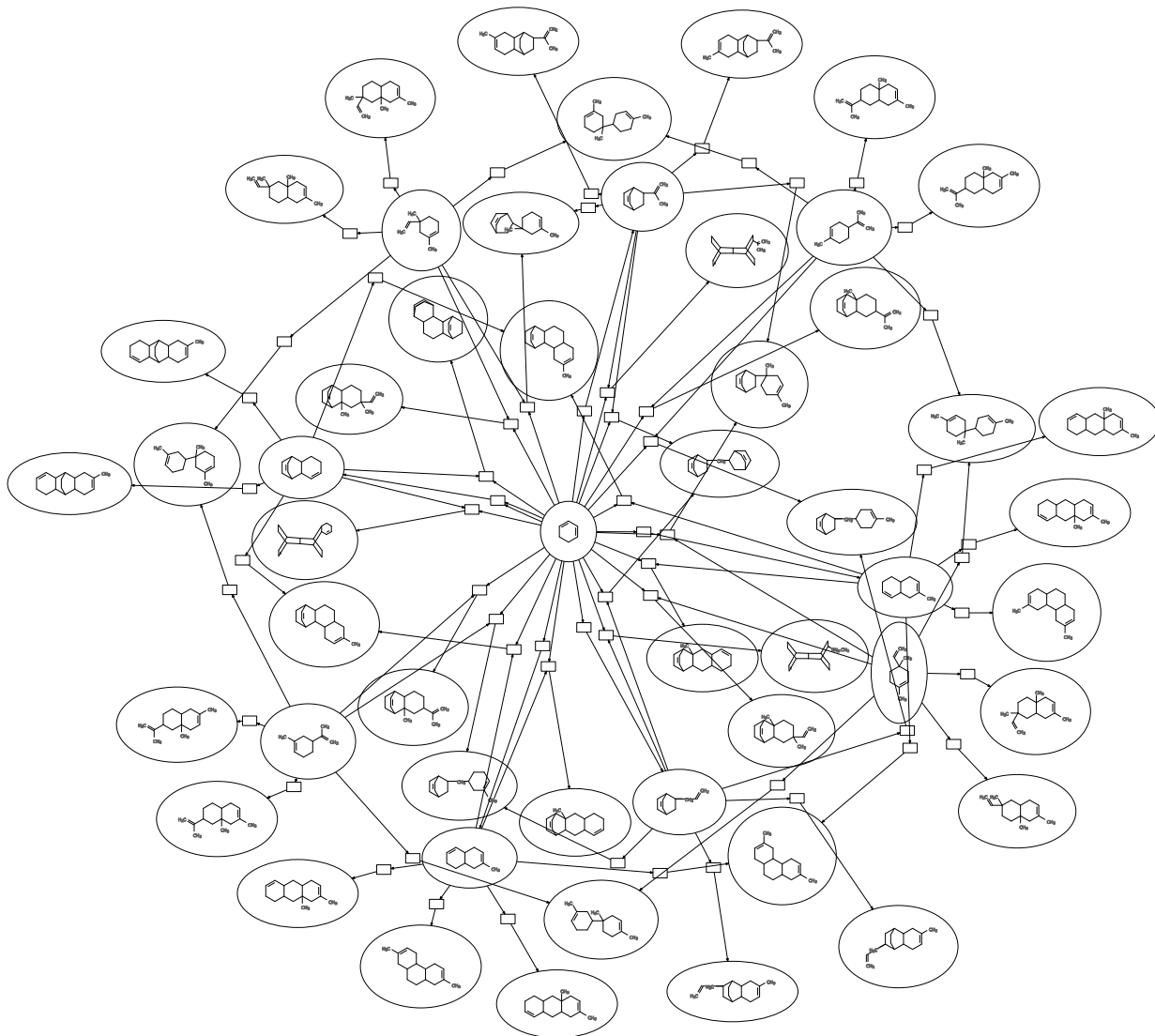
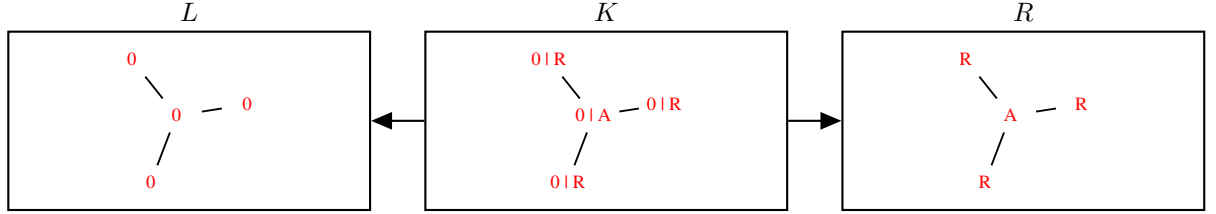


Figure 11: The derivation graph resulting from evaluating the breadth-first expansion strategy $Q_{\text{BFS}} = \text{addSubset}[\{\text{isoprene}, \text{cyclohexadiene}\}] \rightarrow \text{repeat}[Q_p, 2]$ (on an empty graph state). To minimize clutter, the vertex with isoprene and the corresponding edges are not shown.

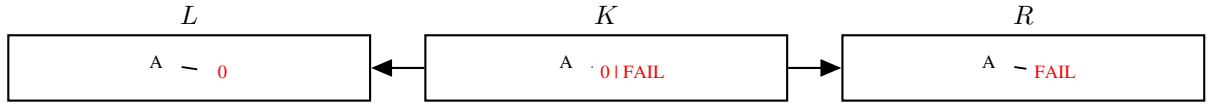
B Transformation Rules for the Catalan Game

The following sections contain visualization of the rules used in the strategy to solve a level in the Catalan game. Vertices and edges shown in red are those being changed during transformation. For some vertices the change is only a change of label. The label in the context graph, K , is for those in the format “L | R” with L and R being the label in the left and right side of the rule.

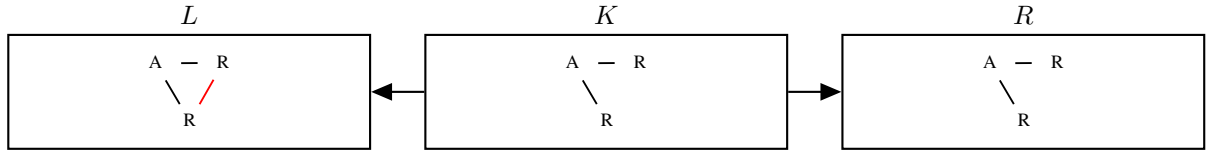
B.1 mark



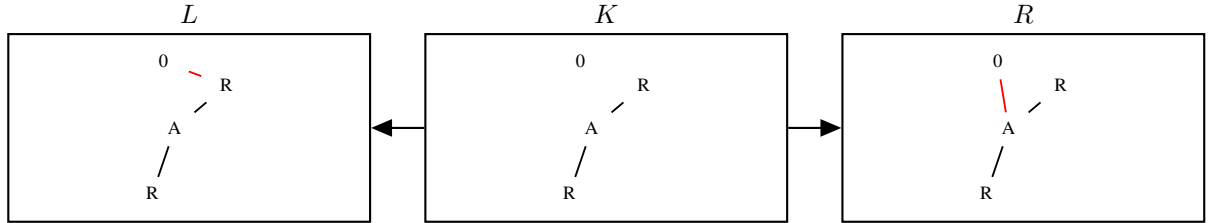
B.2 markForFail



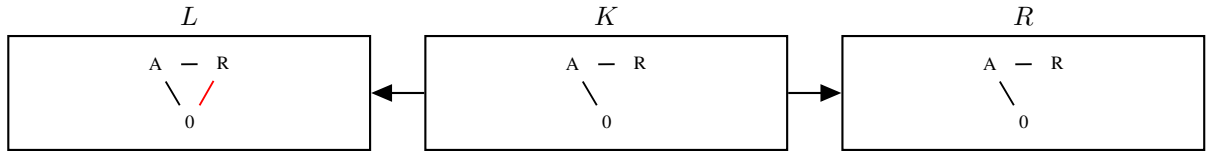
B.3 removeInterR



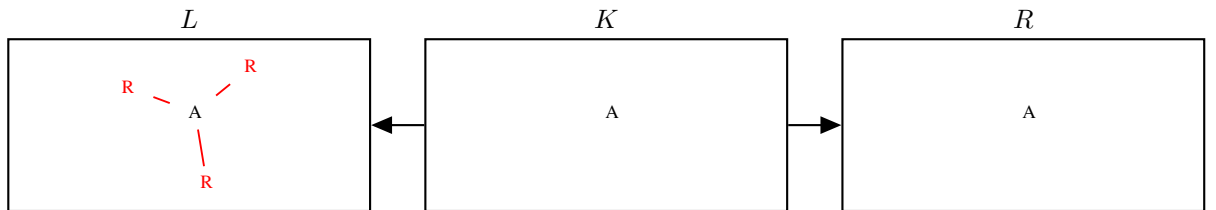
B.4 reattachExternal



B.5 removeAttached



B.6 removeR



B.7 unmark

